Real-time Placement of Labels on a Geographical Map

Mo'tassem Al-arydah (University of Ottawa)
Kamel Bentahar (University of Oxford)
Alain Hertz (École Polytechnique de Montréal)
Odile Marcotte (CRM and UQÀM)
Asma Mdimagh (École Polytechnique de Montréal)
Katayoon Moazzami (École Polytechnique de Montréal)
Patrick St-Louis (GIRO Inc.)
Duc Minh Vu (Université de Montréal)

August 21, 2009

Definition of the Classical Placement Problem

Statement
Given labels (used to label map entities) and sets of positions for these labels, place labels on the map so that it be legible and contain as much information as possible.

The "legibility" requirement means (among other things) that

- a label must be placed fairly close to the corresponding entity (i.e., the label position must be of "high quality"), and
- there is not much overlap between the labels.

An Attempt to Formalize the Classical Problem

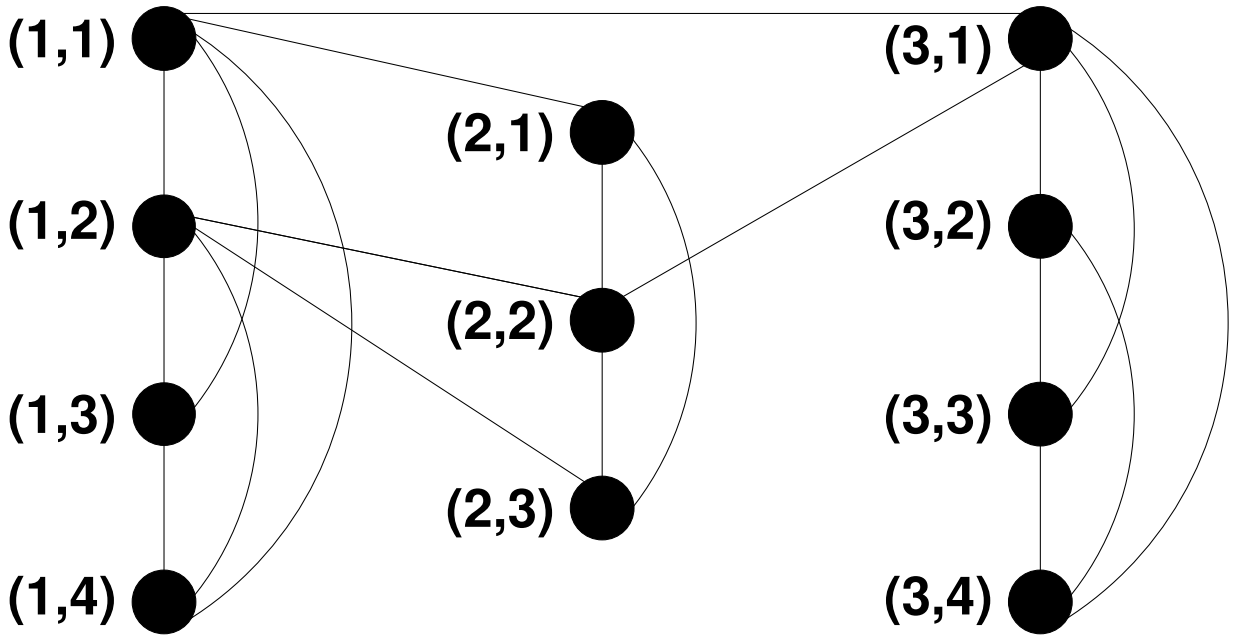
Let G be an undirected graph (the *conflict graph*) defined as follows:

- the vertices of G are of the form (i, j) (or ij for short), where i denotes a label and $j \in P_i$ one of the positions for label i,
- there is an edge between ij and $i\ell$ for any pair $\{j, \ell\}$ of distinct positions, and
- there is an edge between ij and $k\ell$ if the labels i and k are different and the positions j and ℓ overlap.

Each label i has a priority p_i. The problem is then to choose a *stable set* S of vertices of G such that

- S contains exactly one vertex of the form ij for each i, and
- the quantity $\sum_{ij \in S} p_i$ is maximum.

An Illustration of the Conflict Graph



Another Formulation of the Classical Problem

Assume now that we allow overlaps.

- The cost of vertex ij, denoted by c_{ij}, is defined as $p_i \cdot \mathit{weight}(j)$, where $\mathit{weight}(j)$ is a measure of the quality of position j.
- The cost of edge $\{ij, k\ell\}$, denoted by $c_{ij,k\ell}$, is defined as

$$\max\{c_{ij}, c_{k\ell}\} \cdot (c_{ij} + c_{k\ell}) \cdot \text{ percentage of overlap.}$$

- The cost of a subgraph is the sum of the costs of all its edges.

The problem consists of choosing a subset S such that

- S contains exactly one vertex of the form ij for each i, and
- the cost of the subgraph induced by S is minimum.

A Mathematical Programming Formulation

The (binary) variables are the x_{ij} and the $y_{ij,k\ell}$.

- x_{ij} equals 1 if and only if the vertex (i, j) belongs to S.
- $y_{ij,k\ell}$ equals 1 if and only if (i, j) and (k, ℓ) belong to S and there is a conflict between those two vertices.

$$\text{minimize } \sum c_{ij,k\ell}\, y_{ij,k\ell}$$

$$\text{such that } y_{ij,k\ell} \geq x_{ij} + x_{k\ell} - 1 \text{ for each edge}$$

$$\sum_j x_{ij} = 1 \text{ for each } i$$

$$x_{ij} \in \{0, 1\}, y_{ij,k\ell} \in \{0, 1\}$$

Features of the GIRO problem

GIRO develops and sells software for helping clients solve their routing problems. The software includes a display module.

- The user may modify interactively the solution produced by the software.
- The user selects the labels he wants the software to display.
- The display module takes some time to compute an alternative (i.e., non preferred) position for a given label (it has to take into account the conflicts between labels and entities, for instance).

Hence the label placement algorithm must find a good placement while sending as few requests as possible to the display module.

The user, the display module, and the algorithm

- The user selects the labels he wants displayed (he may have modified some entities beforehand).
- The display module gathers the requested data.
- The algorithm obtains label priorities and tries to compute a good placement.
- If necessary, the algorithm requests more information from the display module, i.e., information on overlaps in the current solution or on alternative positions.
- The algorithm produces a placement for the display module.

Outline of the Proposed Solution

The main idea is the decomposition of the map into relatively small areas, so that few requests are made to the display module and the conflict graph is of moderate size.

1. Preprocessing stage (creation of the "grid")
2. Selection of a partial grid
3. Request of t_1 positions for each label whose entity lies within the block (for each block)
4. Computation of a solution for each block (using an exact algorithm or a heuristic followed by a post-optimization procedure)
5. Identification of the labels that are poorly placed
6. Request (for each label) the next t_2 positions

The sub-algorithm consisting of the last two steps can be iterated.

Grid: Preprocessing

Block: connected region defined by geographical features (rivers, roads, . . .)

Grid: The attack of labels!

We may infer the density of the labels from the density of the “entities.”

Grid: Clustering blocks

If a set of blocks are not dense then we merge them to allow more freedom in placing the labels.

Grid: Clustering blocks

If a set of blocks are not dense then we merge them to allow more freedom in placing the labels.

Grid: Clustering blocks

If a set of blocks are not dense then we merge them to allow more freedom in placing the labels.

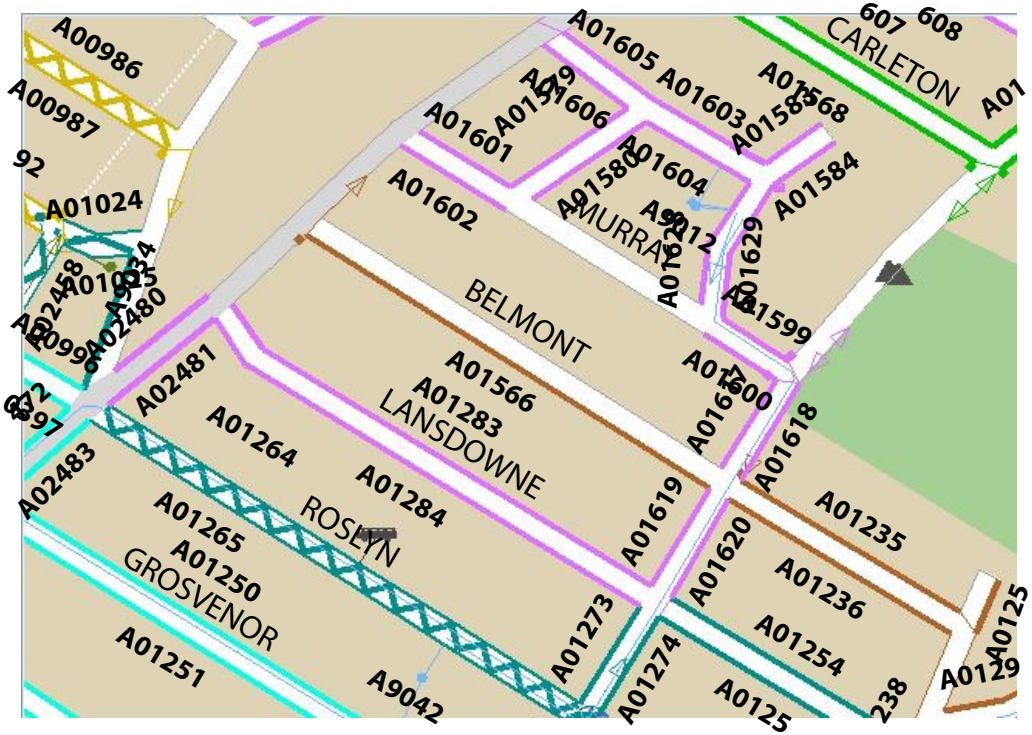
Grid: Clustering blocks

If a set of blocks are not dense then we merge them to allow more freedom in placing the labels.

Grid: Clustering blocks

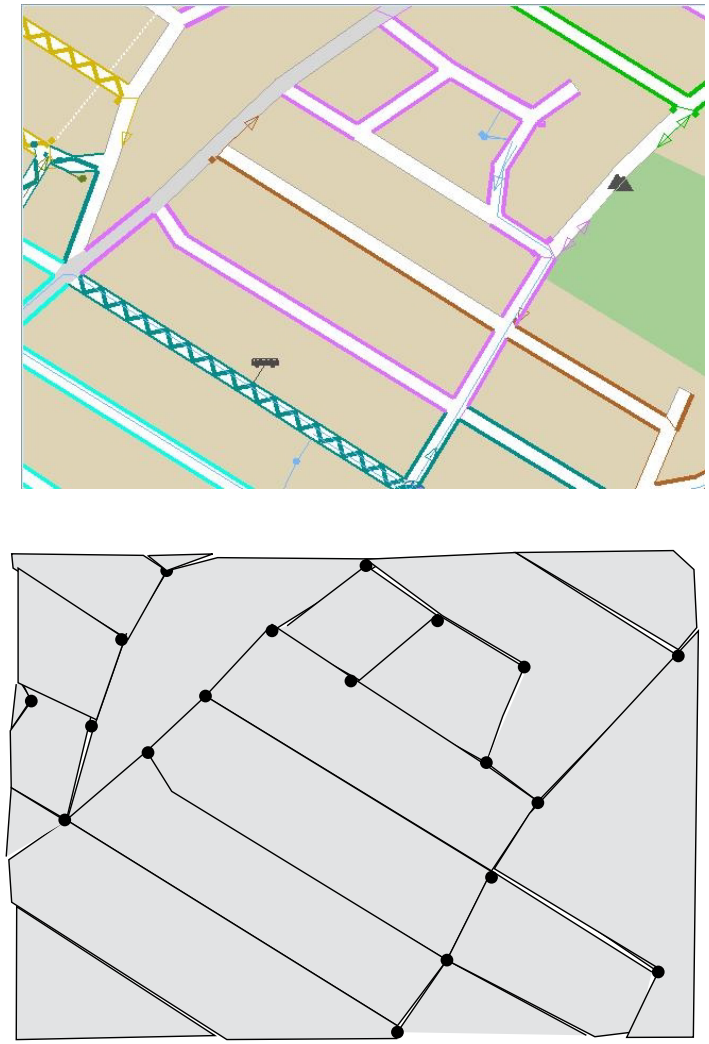
k-**clustering** algorithms or repeat 2-**clustering** on dense regions.

Figure no 1



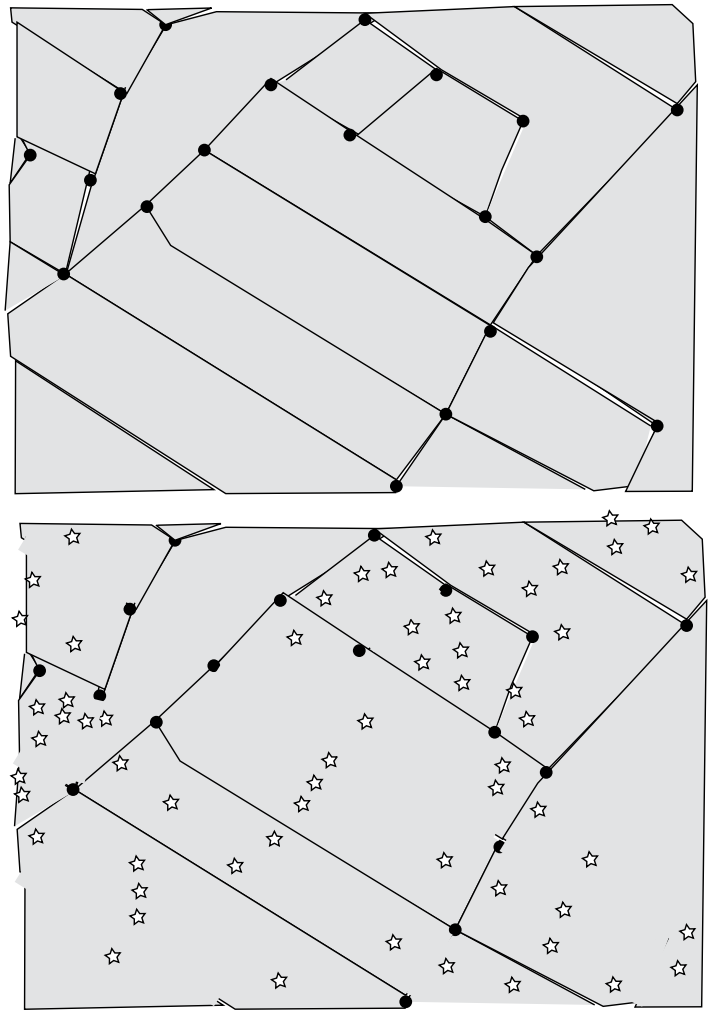
A solution provided by the software at GIRO

Figure no 2



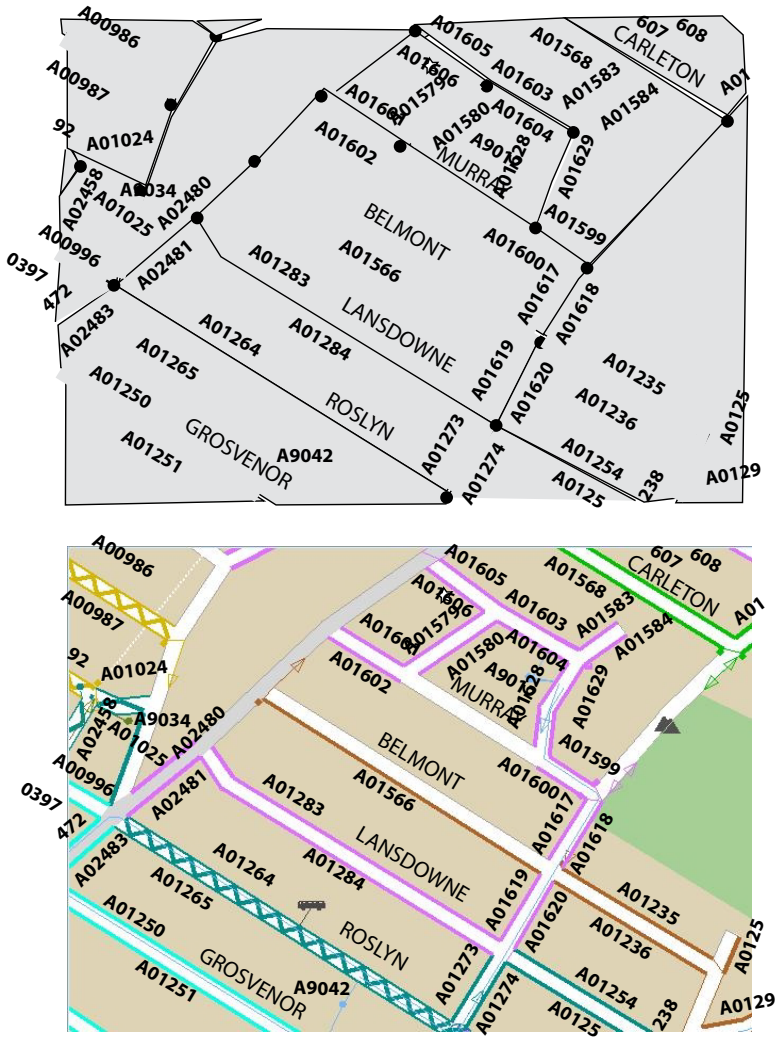
The associated map with its decomposition into blocks

Figure no 3



Each label that has to be positioned is represented by a star
at the favorite location
Blocks are merged if their total number of stars is not too big
(at most 10 in this example)

Figure no 4



Each block is optimized independently

Figure no 5

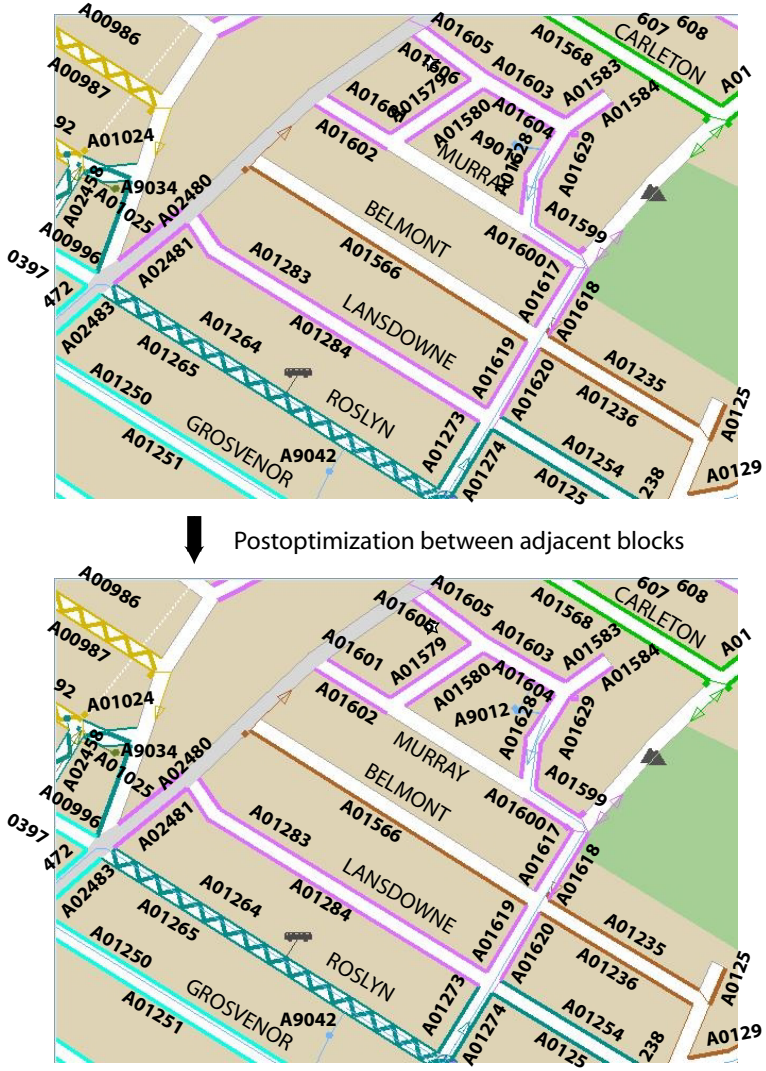
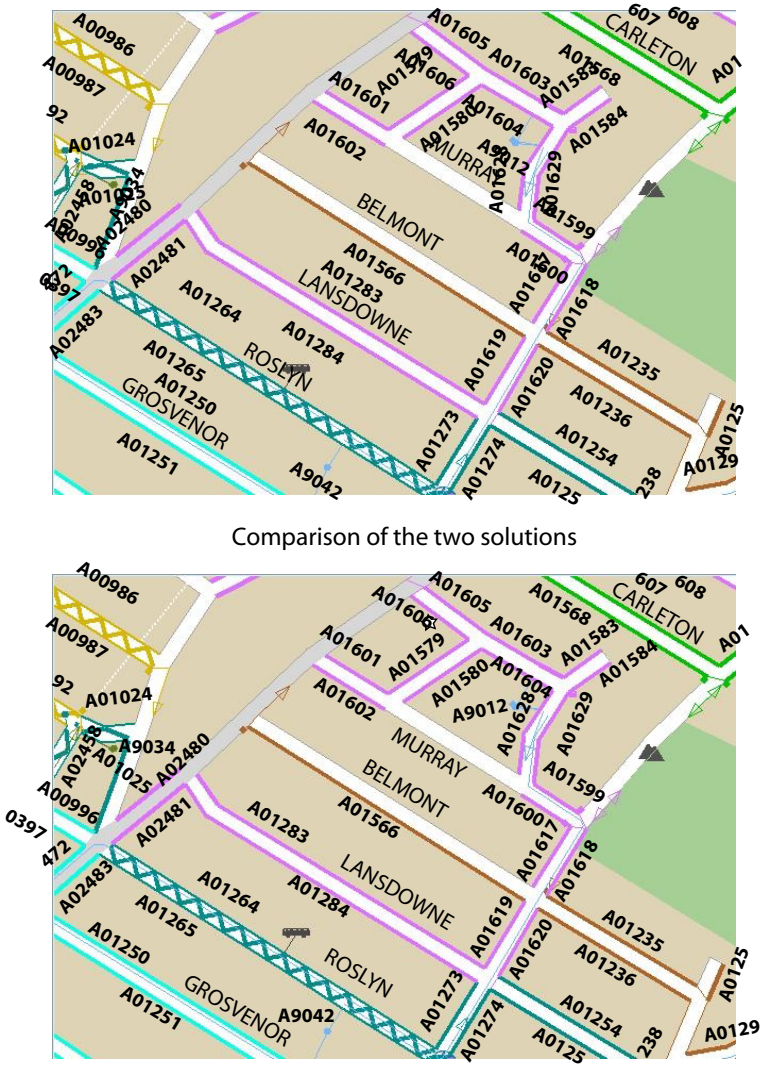

A00986
A00987
92
A01024
A02458
A9034
A01025
A02480
A00996
0397
472
A02481
A02483
A01250
A01251
A9042
A01602
A01605
A01606
A01579
A01568
A01583
A01584
607
608
CARLETON
A01580
A01604
A9012
A01628
MURRAY
A01629
A01599
BELMONT
A01566
A016007
A01617
A01618
A01283
LANSDOWNE
A01284
A01264
A01265
ROSLYN
A01619
A01620
A01235
A01236
A01273
A01274
A01254
A01125
A0129
238
Postoptimization between adjacent blocks
A01601

Figure no 6



Comparison of the two solutions

Step 3: Placing Labels Within a Block

Here are some ideas for solving the placement problem within a block.

- A Tabu Search Heuristic proposed by Yamamoto et al. (2002)
- A GRASP (Greedy Randomized Adaptive Search Procedure) proposed by Cravo et al. (2007)
- Resolution of the integer program (see the second formulation above)
- Dual ascent procedure (adapted from Erlenkotter's procedure for the plant location problem)

Step 4: Identification of Poorly Placed Labels

Within a given block, let S denote the assignment of positions to labels within the block, i.e., the set

$$\{(i,j) \mid i \text{ is a label and } j \text{ its position}\}.$$

As above, let $c_{ij,k\ell}$ denote the cost of edge $\{ij, k\ell\}$. Here are some criteria for "selecting" poorly placed labels.

- Choose a vertex (i,j) maximizing $\sum_{(k,\ell)\in S} c_{ij,k\ell}$ and request more positions from the display module.
- Choose a subset S' of S such that for each (i,j) in S, $\sum_{(k,\ell)\in S} c_{ij,k\ell}$ is at least a certain threshold; then request more positions for each (i,j) in S'.
- Find a maximum cardinality stable set in S and request more positions for the vertices in S that do not belong to this stable set.